Exercise: The LSF batch system

To do the hands-on exercises, you need to connect to the LSF front-end node for this course first. Use either

- SSH: 'ssh hpc2019.gbar.dtu.dk', and then start a 'linuxsh' from the command prompt.
- ThinLinc: start the ThinLinc client, and connect to 'hpc2019.gbar.dtu.dk'. From the ThinLinc session, you need to start an 'xterm-appnode' from the 'Applications -> DTU' menu.

For all scripts below, use `'/bin/sleep 60'` (or a longer period, like 100 or 120 seconds) as the command to run, and use 'hpcintro' as the queue name (except where stated otherwise).

1. Write a simple job script, like the one shown in the lecture and submit it.
   (a) Check the status with bstat and/or bjobs. Use 'man bjobs', to get information about the options.
   (b) You can add a walltime limit to the script. Can you see that limit in the bstat or the bjobs output?
2. Write a job script that sends you notifications when the job starts and ends - see 'man bsub' for the details. Take a look at the job summary, i.e. which information can you retrieve from that?
   (a) To test, increase the period in the sleep command to be longer than the walltime limit, and submit the job again. What happens?
3. The default 'hpc' queue has nodes of different type, e.g. CPUs. The CPU type can be requested as a feature in a command script.
   (a) Use the 'nodestat' command to check which CPU types are available in the 'hpc' queue, and then submit a job script that requests one of the types. See the "Batch Jobs under LSF 10" webpage for information how to do that.
   (b) Add the necessary commands to your job script, to print the CPU type - and check in the job output that your job did indeed run on a node with the requested feature. Note: there is slight difference in the type request and the type variable: the variable contains a '-', while LSF uses a '_'.

   The next steps are a preparation for week 2, where we want to submit multi-core jobs to the batch system:
4. Write a job script that requests 1 node and 4 cores. How can you achieve that?
5. Write a job script, requesting one node and 16 cores. Does it run? If the job doesn't start, use 'bjobs -p' to check for the reason.
6. Write a job script, requesting one node and 32 cores. Does it run? If the job doesn't start, use the 'bjobs -p' command to check for the reason.

   Clean up:
7. Check all your submitted jobs with 'bstat' again. If there are any left, that still are in status 'PEND', please remove them with 'bkill JOBID' (the JOBID is the number in the first column of the 'bstat'/'bjobs' output).

Hints: to get the informations needed, you should use the 'man' command and take a look at the DTU Computing Center webpages `https://www.hpc.dtu.dk/?page_id=2534`